MDA Overview

Bill Wood
Overview

- Introduction
- Concepts
- Analysis of Current Work
- Connections
- Next Steps
- Conclusions
Introduction

• Paradigm shift: from programmers using programming language to modelers using UML
• Internal scouting paper
• Scope of investigation: Examination of published accounts (journal articles, OMG web site, before spring 04) – not interviews with practitioners or usage
Introduction

• Multiple models of the system
  – CIM, PIM, PSM, Implementation
  – Build a “business model” first and reuse it

• Standards based
  – Core profiles: UML, MOF, CWM
  – Other: QVT, XML

• Tool driven
  – UML, automated transformation

• Focused on building an implementation
OMG Umbrella

- OMG supports a suite of standards
  - CORBA, UML, MDA,
  - And emerging standards ASL, QVT etc.
- CORBA standards require a PIM in UML and a PSM in IDL
Concepts

• UML
• MDA
• Roles
Concepts - UML

• MDA emphasizes views that lead to implementations directly
  – Classes, state machines
• Other views are not encouraged
• You don’t have to use UML to create a PIM, but it’s the de facto modeling notation because of the tool support.
MDA Transformation

Note that the PSMs are platform specific

Based on Kleppe et al., MDA Explained, 03
MDA Roles

• PIM analyst
  – Knowledge of business rules

• PSM creator
  – Knowledge of target platform
  – Knowledge of transformation definitions

• Transformation definition developer
  – Works for tool vendor or separate group
  – Deep knowledge of source and target languages
  – Deep knowledge of applicable patterns
Analysis of Current Work

- Adoption Thermometer
- MDA Promises
- MDA Challenges
MDA Adoption Thermometer

- MDA is at the peak position now.
- Many technologies come apart on the downslope of the curve.

From Gartner Associates
MDA Promises\(^{(1)}\)

- **Productivity**
  - Save time writing less code
  - Reduced “development test” effort of PIM

- **Quality**
  - generated code has fewer bugs

- **More accurate documentation**
  - Rich in details, always up-to-date
  - Traceability across models
MDA Promises

- Interoperability
  - Bridges are included in models
- Portability
  - Easier to port a model than to port code
MDA Process – Challenges

• Writing transformation rules is very difficult
  – Mostly an MDA tool vendor responsibility, but architects should understand what transformation the tool performs
  – If the tool transformation is not satisfactory, it’s unlikely that a developer will be able to redefine it
  – Transformation language not a standard yet

• Architects are not used to creating PIM-level design that is complete and consistent
  – SE culture: syntactic and semantic discipline only in source code, not models
MDA Process – Challenges

• UML today:
  – Good for structural aspects
  – Awkward for behavioral aspects

• Action Semantics Languages (xUML)
  – Not widely used
  – Suitable for some domains (e.g. embedded), overkill for others (e.g. MIS)
MDA Process – Challenges

• Tool support is crucial
  – Build, buy, or customize
  – PIM to PSM: barely available
  – PSM to code and PIM to code:
    • Many CASE tools have black-box generators
    • Often behavior has to be added manually to the code as only stubs are generated
  – Transformation language is tool specific
MDA Process – Challenges

- Tool support as a limiting factor
  - limited to target platforms supported by a particular vendor
  - limited to the transformations supplied by a vendor
  - may be able to write new transformations or get a vendor contractor to help
MDA Process – Challenges

Three big architectural questions

– How does one decide which transformations to apply in order to meet quality requirements?
– How does one know if the quality requirements are met?
– Can all qualities be built-in by transformation (i.e., without changing the PIM)?

MDA is unclear on these points.
MDA Process – Challenges

• Skill levels and training
  – PIM creators vs. PSM creators vs. implementers.
  – Is it feasible/desirable to turn every developer into a modeler?

• How to assess new requirements and change requests:
  – Do they affect the code, the PSM, or the PIM?
  – Harder to locate the model/code/ transformation rule that is affected by a change request.

• Connection between CIM and PIM is tenuous at best, so the path from requirements to design is muddy
Connections

- Software architecture
- Fault Tolerance variations
- Product Lines
- Importance DoD
- Challenges
Software Architecture

- Software architecture focus

- Communication between stakeholders
- Evaluation of quality attributes
- Blueprint for construction
- Few useful design guidelines
Fault Tolerance Variations-1

• Single PIM may be the basis for PSMs on many weapons platforms with different characteristics

• Health Monitoring
  – Heartbeat, ping/echo, BIT
  – Centralized or Distributed

• Style
  – Active; passive: warm, standby, cold
Fault Tolerance Variations-1

• Data Integrity
  – Lose transient requests / No lost Requests
  – Active replication / Checkpoint and recovery
  – Reliable Ordered Multicast Protocol / Asynchronous uni-cast

• Failure transparency to client’s request
  – Middleware transparent / Aliasing Scheme / Client uses library function

• States and Modes
  – Weapons platform host startup / Local Startup
  – Tactical / Training Mission / Mission Rehearsal
Product Lines

- PIM is the basis of a production system for software product lines. Supports multiple variants (PSMs) and implementations.
- The CIM can be used to model the product line requirements.
  - Issue: Quality requirements
  - CIM is given short shrift in the literature
Importance- DoD

- DoD relies on commercial computing platforms as the basis for system development
- DoD relies on commercial toolsets for architecting and documenting the system
- DoD requires software with similar capabilities to execute on different computational platforms on different weapons platforms
- Different weapons platforms have different quality attribute conditions (performance, availability, security, etc)
Challenges - for big systems

1. How to successfully connect the system and software architectures
2. How to define the “platform” layers in going from PIM to PSM
3. How to introduce quality attributes into the platform layers and the transformations
4. Incorporation of legacy platforms and software in early migration steps and their replacement in later steps
5. How to change toolsets
Importance- General

• Raises the level of abstraction
• Improve life-cycle costs of systems
• Prevent “throwaway” models
• Tool vendors are focusing on MDA
• Reuse of platform independent models
Next Steps

• Survey of user experiences, tool support
• Case study
• Incorporating evaluation of quality attributes to MDA
• Incorporating design guidance to MDA
Conclusions

• MDA is a good idea
  – not yet supported by mature technology
  – still to demonstrate routine success

• Focuses on models for constructing software and disregards software architecture for
  – communicating understanding between stakeholders
  – evaluating quality attributes